GS/OS MS-DOS File System Translator External ERS

Version 0.04

By Greg Branche

REVISION HISTORY

Date	Version	Who	Description
05/13/92	0.01	GAB	Initial release.
05/26/92	0.02	GAB	Typographical corrections.
8/13/92	0.03	DAL	The Map Table now contains auxiliary type values.
10/12/92	0.04	GAB	Removed restriction on support of resource forks.

Table of Contents

Introduction

This document describes the operation and limitations of the GS/OS MS-DOS File System Translator (MS-DOS FST). It assumes familiarity with the structure and usage of GS/OS and of the MS-DOS File System.

Limitations

The function of a File System Translator is to provide application programs with transparent access to files from any file system using standard GS/OS system calls. FST's provide only the file access capabilities of GS/OS. They do not implement capabilities or system calls supported by other file systems. Accordingly, the MS-DOS FST:

- supports resource forks on files using the same method that the Macinsoth PC Exchange Control Panel does
- maps the file attributes of a file into their equivalent GS/OS access attributes
- is a read-only implementation (in version 1.0)

This last limitation imposes some obvious restrictions on the GS/OS system calls which require writing data to the disc. These calls will always return a write-protect error after identifying that the file or device requested is present and is in MS-DOS format.

The following lists all the GS/OS system calls supported by the MS-DOS FST; those in bold type perform the indicated function, those in plain type will always return an error (with the exception of Flush; see the call description):

Call #	Name	Call #	Name
$01	Create	**$14**	**Close**
$02	Destroy	$15	Flush
$04	ChangePath	**$16**	**SetMark**
$05	SetFileInfo	**$17**	**GetMark**
$06	**GetFileInfo**	$18	SetEOF
$08	**Volume**	**$19**	**GetEOF**
$0B	ClearBackupBit	**$1C**	**GetDirEntry**
$10	**Open**	**$20**	**GetDevNum**
$12	**Read**	$24	Format
$13	Write	$25	EraseDisk
		$33	**FSTSpecific**

File Types

MS-DOS does not provide a file typing mechanism. This is potentially very limiting since most applications select a particular file type as a filter when calling the standard file tools. Therefore, files from an MS-DOS disc would never be selectable.

The MS-DOS FST provides a partial solution to the problem:

The FST searches a translation table for a matching file name extension. If it finds a match , it returns the associated file type and auxiliary type to the caller. For instance, the file "ABC.TXT" will normally be assigned a file type $04 (text) because of the suffix ".TXT". The MS-DOS FST maintains a table of suffixes and their associated file types and auxiliary types. The FSTSpecific calls allow for modification of this table. The default table contains the following entries:

Extension	GS/OS File Type	GS/OS Auxtype	Description
.TXT	$04	$0000	text file
.BAT	$04	$0000	batch file
.BIN	$06	$0000	binary file
.ASC	$04	$0000	ASCII text file
.C	$04	$0000	C language source code
.H	$04	$0000	C header file
.PAS	$04	$0000	Pascal language source code
.ASM	$04	$0000	assembly language source code
.LST	$04	$0000	listing file
.COB	$04	$0000	COBOL language source code
.FOR	$04	$0000	FORTRAN language source code
.DOC	$04	$0000	documentation file
.SRC	$04	$0000	source code file
.GIF	$C0	$8006	Graphics Interchange Format file

System Calls

This section gives some general information about using the MS-DOS FST and describes the parameters for each of the calls. Italicized text in the call parameter descriptions indicates areas where the MS-DOS FST differs from the descriptions given in the document entitled "GS/OS System Calls."

The MS-DOS directory structure only stores one date and time value. The FST interprets it as a "modified" date and time. Any calls that normally return a creation date and time will return zeros in the associated parameter positions of the call.

The MS-DOS file system has no direct support for forked files (e.g., a separate data and resource fork). A convention has been defined, though, by the Macintosh PC Exchange (PCX) product. When PCX creates a file containing a resource fork, it creates a subdirectory named "RESOURCE.FRK" at the same directory level as the original file, and then creates a file within the "RESOURCE.FRK" subdirectory having the same name as the original file. It is this file that is the repository for the resource fork of the file (with the original file containing the data fork of the file). The MSDOS FST has knowledge of this convention and properly interprets the file information for forked files stored using this method.

The MS-DOS file system stores a file attribute byte in the directory entry for each file. This is similar to the GS/OS access attributes. The FST translates the file attributes as follows:

File Attribute	GS/OS interpretation
Archive bit set	Bit 5 (the "backup" bit) of the access attributes will be set.
Subdirectory bit set	File type will be returned as $000F.
Volume Label bit set	Used internally by the FST to apply a volume name to the disk.
System File bit set	No special action.
Hidden File bit set	Bit 2 (the "Invisible" bit) of the access attributes will be set.
Read-Only File bit set	Bits 7, 1, and 0 (the Delete, Write-enable, and Rename) of the access attributes will be **cleared** (i.e., the file is "locked").

Create ($01)

This call will always return one of the following errors:

$04	parameter count out of range
$10	device not found
$27	I/O error
$2B	write-protected
$2F	device offline
$40	invalid pathname syntax
$44	path not found
$45	volume not found
$46	file not found
$52	unsupported volume type

$53	invalid parameter
$57	duplicate volume
$58	not a block device
$59	invalid class

Destroy ($02)

Always returns an error. Same as Create call.

ChangePath ($04)

Always returns an error. Same as Create call.

SetFileInfo ($05)

Always returns an error. Same as Create call.

GetFileInfo ($06)

Description:

This call returns certain file attributes of an existing open or closed block file.

Class 0 Parameters:

Offset	Label	Description
$00-$03	pathname	input long word pointer: Points to a class 0 string representing the pathname of the file whose file information is returned to the caller.
$04-$05	access	output word value: bits 15-8 reserved, must be zero bit 7 D=0, destroy disabled; D=1, destroy enabled bit 6 RN=0, rename disabled; RN=1, rename enabled bit 5 B=0, backup not needed; B=1, backup needed bits 4-3 reserved, must be zero bit 2 I=0, visible; I=1 invisible bit 1 W=0, write disabled; W=1, write enabled bit 0 R=0, read disabled; R=1, read enabled
$06-$07	file_type	output word value: *$000F if a directory, else $0000 ("unknown") unless the file name matches an entry in the file type mapping table. See FSTSpecific calls.*
$08-$0B	aux_type or total_blocks	output long word value: Zero, unless the (non-directory) filename matches an entry in the file type mapping table. See FSTSpecific calls.
$0C-$0D	storage_type	output word value: $01 = standard file (no resource fork file) $05 = extended file (resource fork file present) $0D = subdirectory $0F = volume directory
$0E-$0F	create_date	output word value: *Always 0.*
$10-$11	create_time	output word value: *Always 0.*
$12-$13	mod_date	output word value:

		bits 15-9 year (1=1901, 2=1902, ...) bits 8-5 month (1=January, 2=February, ...) bits 4-0 day (1-31)
$14-$15	mod_time	output word value: bits 15-13 0 bits 12-8 hour (0-23) bits 7-6 0 bits 5-0 minute (0-59)
$16-$19	blocks_used	output long word value: For a standard file, this field returns the total number of blocks used by the file. For a subdirectory file this field returns the number of blocks occupied by the subdirectory. For a volume, this field returns the total number of blocks used for all purposes on the volume. *(This is always the same as total_blocks).*

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=2)
$02-$05	pathname	input long word pointer: Points to a class 1 string representing the pathname of the file whose file information is returned to the caller.
$06-$07	access	output word value: bits 15-8 reserved, must be zero bit 7 D=0, destroy disabled; D=1, destroy enabled bit 6 RN=0, rename disabled; RN=1, rename enabled bit 5 B=0, backup not needed; B=1, backup needed bits 4-3 reserved, must be zero bit 2 I=0, visible; I=1 invisible bit 1 W=0, write disabled; W=1, write enabled bit 0 R=0, read disabled; R=1, read enabled
$08-$09	file_type	output word value: *$000F if a directory, else $0000 ("unknown") unless the file name matches an entry in the file type mapping table. See FSTSpecific calls.*
$0A-$0D	aux_type	output long word value: Zero, unless the (non-directory) filename matches an entry in the file type mapping table. See FSTSpecific calls.
$0E-$0F	storage_type	output word value: $01 = standard file

		$05 = extended file (resource fork file present) $0D = subdirectory $0F = volume directory
$10-$17	create_date_time	output double long word value: *Always 0.*
$18-$1F	mod_date_time	output double long word value: Value of the file's creation date and time attributes: byte 00 - second (0-59) byte 01 - minute (0-59) byte 02 - hour (0-23) byte 03 - year (year-1900) byte 04 - day (0-30) byte 05 - month (0-11) byte 06 - null byte 07 - day of week (1-7, 1=sunday)
$20-$23	option_list	input long word pointer to output (default null): Points to a data area consisting of a two-byte length field followed by a two-byte output length field, followed by space for the output data. The length field is an input giving the total length of the data area including the length word itself. On output, the MS-DOS FST sets the output length field to a value giving the number of bytes of space required by the output data, excluding the length words. The MS-DOS FST will not overflow the available output data area. If the data area is too small, the caller may reissue the call after allocating a new output buffer with size adjusted to output length + 4.
$24-$27	eof	output long word value: For a standard file, this field gives the number of bytes that can be read. For a volume directory or a subdirectory, this field will be 0.
$28-$2B	blocks_used	output long word value: If the pathname parameter specifies a standard file or subdirectory, this field returns the total number of blocks used by the file. If the call specifies a volume directory, this field is undefined.
$2C-$2F	resource_eof	output long word value: If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the end-of-file value for the resource fork file associated with the named file.
$30-$33	resource_blocks	output long word value:

If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the number of blocks used by the resource fork file associated with the named file.

Errors:

$04	parameter count out of range
$10	device not found
$27	I/O error
$40	invalid pathname syntax
$44	path not found
$45	volume not found
$46	file not found
$4B	unsupported storage type
$52	unsupported volume type
$53	invalid parameter
$58	not a block device

Volume ($08)

Description:

Given the name of a block device, this call returns the name of the volume mounted in the device. Other information about the volume is returned as well.

Class 0 Parameters:

Offset	Label	Description
$00-$03	dev_name	input long word pointer Points to a class 0 string containing the name of a block device.
$04-$07	vol_name	input long word pointer to output string Points to a data area where the MS-DOS FST places a class 0 string containing the volume name of the volume mounted in the device.
$08-$0B	total_blocks	output long word value Total number of blocks contained on the volume.
$0C-$0F	free_blocks	output long word value Total number of unallocated blocks contained on the volume.
$10-$11	file_sys_id	output word value Identifies the file system contained on the volume. The MS-DOS FST will always return a value of $000A. *If the volume is not recognized by any installed FST, GSIOS will return an "unsupported volume type" error ($52).*

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	input word value parameter count (minimum=2)
$02-$05	dev_name	input long word pointer Points to a class 1 string containing the name of a block device.
$06-$09	vol_name	input long word pointer to output string Points to a class 1 output string where the MS-DOS FST places the volume name of the volume mounted in the device.
$0A-$0D	total_blocks	output long word value Total number of blocks contained on the volume.
$0E-$11	free_blocks	output long word value Total number of unallocated blocks contained on the volume.
$12-$13	file_sys_id	output word value Identifies the file system contained on the volume. The MS-DOS FST will always return a value of $000A. *If the volume is not recognized by any installed FST, GS/OS will return an "unsupported volume type" error ($52).*
$14-$15	block_size	output word value This field contains the size, in bytes, of a block.

Errors:

$04	parameter count out of range
$10	device not found
$11	invalid device request
$27	I/O error
$28	no device connected
$2E	disk switched
$45	volume not found
$52	unsupported volume type
$53	invalid parameter
$57	duplicate volume
$58	not a block device

ClearBackupBit ($0B)

Always returns an error. Same as Create call.

Open ($10)

Description:

This call causes the MS-DOS FST to set up an access path to a file. Once an access path is set up, the user may perform file READs and other related operations on the file.

The class 1 OPEN call also optionally returns all the file information returned by the GetFileInfo call.

A file may be opened more than once (limited only by memory) and each open will assign a different reference number.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	output word value A reference number assigned by GS/OS to the access path. All other file operations (READ, CLOSE, etc.) refer to the access path by this number.
$02-$05	pathname	input long word pointer Points to a class 0 string representing the pathname of the file to be opened.
$06-$09	reserved	This field is reserved and must be set to $00000000.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=2)
$02-$03	ref_num	output word value A reference number assigned by GS/OS to the access path. All other file operations (READ, CLOSE, etc.) refer to the access path by this number.
$04-$07	pathname	input long word pointer Points to a class 1 string representing the pathname of the file to be opened.
$08-$09	request_access	input word value (default "as allowed") Specifies the desired access permissions. bits 15-2 reserved, must be zero bit 1 W=1, request write permission bit 0 R=1, request read permission If this field is not included or its value is $0000, the file will be opened read-only (since it's opened by a read-only FST). *If this field is included and its value is anything other than $0000 or $0001, the MS-DOS FST will return an access error ($4E).*
$0A-$0B	resource_number	input word value (default=$0000) This field is meaningful only when pathname specifies an extended file. If the resource fork file associated with the named file is to be opened, the parameter should be set to $0001.
$0C-$0D	access	output word value: Value for the file's access attribute (described under the GetFileInfo call).
$0E-$0F	file_type	output word value *$000F if a directory, else $0000 ("unknown") unless the file name matches an entry in the file type mapping table. See FSTSpecific calls.*

$10-$13	aux_type	output long word value Zero, unless the (non-directory) filename matches an entry in the file type mapping table. See FSTSpecific calls.
$14-$15	storage_type	output word value Value of the file's storage_type attribute. $01=standard file $05 = extended file (resource fork file present) $0D = subdirectory $0F = volume directory
$16-$1D	create_date_time	output double long word value *Always 0.*
$1E-$25	mod_date_time	output double long word value Value of the file's modification date and time attributes. See GetFileInfo call for format.
$26-$29	option_list	input long word pointer to output (default null) Points to a data area consisting of a two-byte length field followed by a two-byte output length field, followed by space for the output data. The length field is an input giving the total length of the data area including the length word itself. On output, the MS-DOS FST sets the output length field to a value giving the number of bytes of space required by the output data, excluding the length words. The MS-DOS FST will not overflow the available output data area. If the data area is too small, the caller may reissue the call after allocating a new output buffer with size adjusted to output length + 4.
$2A-$2D	eof	output long word value For a standard file, this field gives the number of bytes that can be read from the file. For a subdirectory file, this field is 0. For the volume directory, this field is undefined.
$2E-$31	blocks_used	output long word value If the pathname parameter specifies a standard file, this field returns the total number of blocks used by the file. If it specifies an extended file, this field returns the number of blocks used by the file's data fork, even if one is opening the resource fork. If the call specifies a subdirectory or volume directory, this field is undefined.
$32-$35	resource_eof	output long word value

		If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the end-of-file value for the resource fork file associated with the named file.
$36-$39	resource_blocks	output long word value
		If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the number of blocks used by the resource fork file associated with the named file.

Errors:

$04	parameter count out of range
$27	I/O error
$28	no device connected
$2E	disk switched
$40	invalid pathname syntax
$44	path not found
$45	volume not found
$46	file not found
$4B	unsupported storage type
$4E	access not allowed
$50	file is open
$52	unsupported volume type
$58	not a block device

Read ($12)

Description:

This function tries to transfer request_count bytes, starting at the current mark, from the file specified by ref_num into the buffer pointed to by data_buffer. The file mark is updated to reflect the new file position after the read.

Two factors may cause fewer than request_count bytes to be transferred, so the function returns the actual number of bytes transferred in transfer_count. If the MS-DOS FST reaches the end of file before transferring request_count bytes, it stops the read and sets transfer_count to the number of bytes physically transferred. If newline mode is enabled and a newline character is encountered before request_count bytes have been read, the MS-DOS FST stops the transfer and sets transfer_count to the number of bytes physically transferred, including the newline character.

The MS-DOS FST allows reading of directories, but since directories appear to have an EOF value of 0, a read call will *always* encounter the end-of-file before transferring any bytes.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value
		The identifying number assigned to the file by the OPEN call.
$02-$05	data_buffer	input long word pointer
		Points to a memory area large enough to hold the requested data.
$06-$09	request_count	input long word value
		The number of bytes to be read.
$0A-$0D	transfer_count	output long word value
		The number of bytes actually read.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=4)
$02-$03	ref_num	input word value The identifying number assigned to the file by the OPEN call.
$04-$07	data_buffer	input long word pointer Points to a memory area large enough to hold the requested data.
$08-$0B	request_count	input long word value The number of bytes to be read.
$0C-$0F	transfer_count	output long word value The number of bytes actually read.
$10-$11	cache_priority	input word value (default = $0000) Specifies whether or not disc blocks handled by the read call are candidates for caching. $0000 do not cache blocks involved in this read $0001 cache blocks involved in this read if possible

Errors:

$04	parameter count out of range
$27	I/O error
$2E	disk switched
$43	invalid reference number
$4C	eof encountered
$4E	access not allowed

Write ($13)

Returns write-protect error ($2B).

Close ($14)

Description:

This call closes the access path to the specified file, releasing all resources used by the file and ending further access to it.. Memory resident data structures associated with the file are released.

If the specified ref_num is $0000, all files at or above the current system file level are closed.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value The identifying number assigned to the file by the OPEN call. A value of $0000 indicates that all files at or above the current system file level should be closed.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=1)
$02-$03	refnum	input word value The identifying number assigned to the file by the OPEN call. A value of $0000 indicates that all files at or above the current system file level should be closed.

Errors:

$04	parameter count out of range
$27	I/O error
$2B	disk write protected
$2E	disk switched
$43	invalid reference number
$48	volume full
$5A	block number out of range

Flush ($15)

Does nothing. Returns with carry clear.

SetMark ($16)

Description:

This call sets the file mark (the position from which the next byte will be read or to which the next byte will be written) to a specified value. The value may not exceed EOF, the current size of the file.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value The identifying number assigned to the file by the OPEN call.
$02-$05	position	input long word value The value assigned to the mark. It is the position (in bytes relative to the beginning of the file) at which the next read will begin.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=3)
$02-$03	ref_num	input word value The identifying number assigned to the file by the OPEN call.
$04-$05	base	input word value A value that tells how to interpret the displacement field. $0000 set mark = displacement $0001 set mark = eof - displacement $0002 set mark = mark + displacement $0003 set mark = mark - displacement
$06-$09	displacement	input long word value A value used to compute the new value of the mark as described above.

Errors:

$04	parameter count out of range
$27	I/O error
$43	invalid reference number
$4D	position out of range
$5A	block number out of range

GetMark ($17)

Description:

This function returns the current mark for the specified file.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value The identifying number assigned to the file by the OPEN call.
$02-$05	position	output long word value The current value of the mark in bytes relative to the beginning of the file.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum = 2)
$02-$03	ref_num	input word value The identifying number assigned to the file by the OPEN call.
$04-$07	position	output long word value The current value of the mark in bytes relative to the beginning of the file.

Errors:

$04	parameter count out of range
$43	invalid reference number

SetEOF ($18)

This function returns write-protect error ($2B).

GetEOF ($19)

Description:

This function returns the current logical size of a specified file.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value The identifying number assigned to the file by the OPEN call.

$02-$05	eof	output long word value
		The current logical size of the file, in bytes.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum = 2)
$02-$03	ref_num	input word value
		The identifying number assigned to the file by the OPEN call.
$04-$07	eof	output long word value
		The current logical size of the file, in bytes.

Errors:

$04	parameter count out of range
$43	invalid reference number

GetDirEntry ($1C)

Description:

This call returns information about a directory entry in the volume directory or a subdirectory. Before executing this call, the caller must open the directory or subdirectory. The call allows the caller to step forward or backward through file entries or to specify absolute entries by entry number.

Class 0 Parameters:

Offset	Label	Description
$00-$01	ref_num	input word value
		The identifying number assigned to the directory or subdirectory file by the Open call.
$02-$03	flags	Flags that indicate various attributes of the file:
		bit 15 = 0 the file is not an extended file bit 15 = 1 the file has an associated resource fork file
		bits 14-0 reserved for Apple.
$04-$05	base	input word value
		A value that tells how to interpret the displacement field.
		$0000 displacement gives an absolute entry number

		$0001 displacement is added to current displacement to get next entry number $0002 displacement is subtracted from current displacement to get next entry number
$06-$07	displacement	input word value In combination with the base parameter, specifies the directory entry whose information is to be returned. When the directory is first opened, the MS-DOS FST sets the current displacement value to 0. The current displacement value is updated on every GetDirEntry call. If the base and displacement fields are both zero, the MS-DOS FST returns a 2-byte value in the entry_num field that specifies the total number of active entries in the subdirectory. In this case, the MS-DOS FST also resets the current displacement to the first entry in the subdirectory. To step through the directory entry by entry, one should set both the base and displacement fields to $0001.
$08-$0B	name	input long word pointer to output string Points to a class 1 string giving the name of the file or subdirectory.
$0C-$0D	entry_num	output word value The absolute entry number of the entry whose information is being returned. This field is provided so a program can get the absolute entry number even if the base and displacement fields specify a relative entry.
$0E-$0F	file_type	output word value *$000F if a directory, else $0000 ("unknown") unless the file name matches an entry in the file type mapping table. See FSTSpecific calls.*
$10-$13	eof	output long word value The value of the eof field of the directory entry.
$14-$17	block_count	output long word value The value of the blocks_used field of the directory entry.
$18-$1F	create_date_time	output double long word value *Always 0.*
$20-$27	mod_date_time	output double long word value

		The value of the modification date/time field of the directory entry. See get_file_info call for format.
$28-$29	access	output word value Value for the file's access attribute (described under the GetFileInfo call).
$2A-$2D	aux_type	output long word value Zero, unless the (non-directory) filename matches an entry in the file type mapping table. See FSTSpecific calls.
$2E-$2F	file_system_id	output word value File system identifier of the file system on the volume containing the file. *Always = $000A .*

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum = 5)
$02-$03	ref_num	input word value The identifying number assigned to the directory or subdirectory file by the open call.
$04-$05	flags	Flags that indicate various attributes of the file: bit 15 = 0 the file is not an extended file. bit 15 = 1 the file is an extended file (the file has an associated resource fork file). bits 14-0 reserved for Apple.
$06-$07	base	input word value A value that tells how to interpret the displacement field. $0000 displacement gives an absolute entry number $0001 displacement is added to current displacement to get next entry number $0002 displacement is subtracted from current displacement to get next entry number
$08-$09	displacement	input word value In combination with the base parameter, specifies the directory entry whose information is to be returned. When the directory is first opened, the MS-DOS FST sets the current displacement

		value to zero. The current displacement value is updated on every GetDirEntry call. If the base and displacement fields are both zero, the MS-DOS FST returns a 2-byte value in the entry_num field that specifies the total number of active entries in the subdirectory. In this case, the MS-DOS FST also resets the current displacement to the first entry in the subdirectory. To step through the directory entry by entry, one should set both the base and displacement fields to $0001.
$0A-$0D	name	input long word pointer to output string Points to a class 1 output string giving the name of the file or subdirectory.
$0E-$0F	entry_num	output word value The absolute entry number of the entry whose information is being returned. This field is provided so a program can get the absolute entry number even if the base and displacement fields specify a relative entry.
$10-$11	file_type	output word value *$000F if a directory, else $0000 ("unknown") unless the file name matches an entry in the file type mapping table. See FSTSpecific calls.*
$12-$15	eof	output long word value The value of the eof field of the directory entry.
$16-$19	block_count	output long word value The value of the blocks_used field of the directory entry.
$1A-$21	create_date_time	output double long word value *Always 0.*
$22-$29	mod_date_time	output double long word value The value of the modification date/time field of the directory entry. See GetFileInfo call for format.
$2A-$2B	access	output word value Value for the file's access attribute (described under the GetFileInfo call).
$2C-$2F	aux_type	output long word value

		Zero, unless the (non-directory) filename matches an entry in the file type mapping table. See FSTSpecific calls.
$30-$31	file_system_id	output word value File system identifier of the file system on the volume containing the file. *Always = $000A* .
$32-$35	option_list	input long word pointer to output Points to a data area where the MS-DOS FST returns FST specific information related to the file. This is the same information returned in the option list of the OPEN and GetFileInfo calls. This field points to a buffer that starts with a length word giving the total buffer size including the length word. The next word is an output length value which is undefined on input. On output, this word is set to the size of the output data excluding the length word and the output length word. The MS-DOS FST will not overflow the available space specified in the input length word. If the data area is too small, the caller may reissue the call after allocating a new output buffer with size adjusted to output length + 4.
$$36-$39	resource_eof	output long word value: If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the end-of-file value for the resource fork file associated with the named file.
$3A-$3D	resource_blocks	output long word value: If the file is a standard file, this value will be 0. If the file is an extended file, this parameter will contains the number of blocks used by the resource fork file associated with the named file.

Errors:

$04	parameter count out of range
$10	device not found
$27	I/O error
$4A	incompatible file format
$4B	unsupported storage type
$52	unsupported volume type
$53	invalid parameter
$58	not a block device

GetDevNum ($20)

Description:

This call returns the device number (dev_num) of a device identified by device name or volume name. Only block devices may be identified by volume name, and then only if the named volume is mounted. Most other device calls refer to devices by device number.

GS/OS assigns device numbers at boot time. They are a series of consecutive integers beginning with 1. There is no defined relationship between devices and specific device numbers.

Because a device may hold different volumes and because volumes may be moved from one device to another, the device number returned for a particular volume name may change.

Class 0 Parameters:

Offset	Label	Description
$00-$03	dev_name	input long word pointer Pointer to a class 0 string representing the device name or volume name (for a block device).
$04-$05	dev_num	output word value The device reference number of the specified device.

Class 1 Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (minimum=2)
$02-$05	dev_name	input long word pointer Pointer to a class 1 string representing the device name or volume name (for a block device).
$06-$07	dev_num	output word value The device reference number of the specified device.

Errors:

$04	parameter count out of range
$10	device not found
$11	invalid device request
$40	invalid device or volume name syntax
$45	volume not found

Format ($24)

This function returns write-protect error ($2B).

EraseDisk ($25)

This function returns write-protect error ($2B).

FSTSpecific ($33)

Description:

This call controls file type mapping by the MS-DOS FST. It is unique in that it uses a command number as one of its parameters and is actually four different calls. This call is class 1 only.

The four calls are:

Map_Enable	- Enable/Disable file type mapping. Default is enabled.
Get_Map_Size	- Return size in bytes of current map.
Get_Map_Table	- Return current map.
Set_Map_Table	- Replace current map.

The format of a map table is as follows:

Map_Size	length of table, including terminator (word)
Record_0	map record (variable length)
Record_1	map record
:	
Record_n	last map record
00	terminator (zero byte)

Map records consist of a text string (MSBs off) followed by a zero byte followed by a file type byte and and auxiliary file type word. The text string can be any length and can include any legal characters for an MS-DOS file name (text must be upper case, for example). In MPW IIGS assembly, a map table can be defined as follows:

```
map_table       dc.w            end-map_table
                ;Length of table.
                dc.b  '.TXT',$00,$04,$00,$00                    ;Record
0 (.TXT maps to filetype $04, auxtype $0000)
                dc.b  '.TYPE',$00,$7f,$34,$12                   ;Record
1 (.TYPE maps to filetype $7F, auxtype $1234)
                .
                .
                .
                dc.b  $00               ;Terminator.
end
```

Map_Enable Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)
$02-$03	file_sys_id	input word value Must = $000A, the MS-DOS file system id.

$04-$05	command_num	input word value
		$0000 for map_enable
$06-$07	enable	input word value
		$0000 = disable file type mapping. $0001 = enable file type mapping (this is the default state).

Get_Map_Size Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)
$02-$03	file_sys_id	input word value
		Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value
		$0001 for get_map_size
$06-$07	map_size	output word value
		The size, in bytes, of the current map table.

Get_Map_Table Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)
$02-$03	file_sys_id	input word value
		Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value
		$0002 for get_map_table
$06-$09	buffer_ptr	input long word pointer
		Points to a memory area large enough to hold the map table. No checking is done on the size of this buffer!

Set_Map_Table Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)

$02-$03	file_sys_id	input word value Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value $0003 for set_map_table
$06-$09	map_ptr	input long word pointer Points to new map table. As long as there is space in memory for the new table, it will replace the old one. If there is not enough space, an out_of_memory error will be returned and the original table will remain in effect. No validity checking is done on the table.

Errors:

$04	parameter count out of range
$53	invalid parameter
$54	out of memory

$04-$05	command_num	input word value
		$0000 for map_enable
$06-$07	enable	input word value
		$0000 = disable file type mapping. $0001 = enable file type mapping (this is the default state).

Get_Map_Size Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)
$02-$03	file_sys_id	input word value
		Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value
		$0001 for get_map_size
$06-$07	map_size	output word value
		The size, in bytes, of the current map table.

Get_Map_Table Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)
$02-$03	file_sys_id	input word value
		Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value
		$0002 for get_map_table
$06-$09	buffer_ptr	input long word pointer
		Points to a memory area large enough to hold the map table. No checking is done on the size of this buffer!

Set_Map_Table Parameters:

Offset	Label	Description
$00-$01	pcount	parameter count (always = $0003)

$02-$03	file_sys_id	input word value Must = $000A, the MS-DOS file system id.
$04-$05	command_num	input word value $0003 for set_map_table
$06-$09	map_ptr	input long word pointer Points to new map table. As long as there is space in memory for the new table, it will replace the old one. If there is not enough space, an out_of_memory error will be returned and the original table will remain in effect. No validity checking is done on the table.

Errors:

$04	parameter count out of range
$53	invalid parameter
$54	out of memory